zenon
by COPA-DATA

zenon Treiber Handbuch

ISPIP

v.8.20

© 2020 Ing. Punzenberger COPA-DATA GmbH

Alle Rechte vorbehalten.

Die Weitergabe und Vervielfältigung dieses Dokuments ist - gleich in welcher Art und Weise - nur mit schriftlicher Genehmigung der Firma COPA-DATA gestattet. Technische Daten dienen nur der Produktbeschreibung und sind keine zugesicherten Eigenschaften im Rechtssinn. Änderungen - auch in technischer Hinsicht - vorbehalten.

Inhaltsverzeichnis

1 Willkommen bei der COPA-DATA Hilfe 5

2 ISPIP 5

3 ISPIP - Datenblatt 6

4 Treiber-Historie 8

5 Voraussetzungen 8

5.1 PC 8

5.2 Steuerung 8

6 Konfiguration 9

6.1 Anlegen eines Treibers 9

6.2 Einstellungen im Treiberdialog 13

6.2.1 Allgemein 13

6.2.2 Connections 17

7 Variablen anlegen 19

7.1 Variablen im Editor anlegen 19

7.2 Adressierung 23

7.3 Treiberobjekte und Datentypen 24

7.3.1 Treiberobjekte 24

7.3.2 Zuordnung der Datentypen 27

7.4 Variablen anlegen durch Import 28

7.4.1 XML Import 28

7.4.2 DBF Import/Export 29

7.5 Kommunikationsdetails (Treibervariablen) 35

8 Treiberspezifische Funktionen 41

8.1 Statusmeldungen 41

8.2 Analogwerte 42

8.3 Kommandos 42

8.3.1 Sollwert für Kommando 43

8.3.2 Unterstützte Befehle 43

8.3.3 Treiberspezifische Funktionen 51

9 Funktion Treiberkommandos **51**

10 Fehleranalyse **57**

10.1 Analysetool 57

10.2 Treiberüberwachung 58

10.3 Checkliste 60

1 Willkommen bei der COPA-DATA Hilfe

ZENON VIDEO-TUTORIALS

Praktische Beispiele für die Projektierung mit zenon finden Sie in unserem YouTube-Kanal (https://www.copadata.com/tutorial_menu). Die Tutorials sind nach Themen gruppiert und geben einen ersten Einblick in die Arbeit mit den unterschiedlichen zenon Modulen. Alle Tutorials stehen in englischer Sprache zur Verfügung.

ALLGEMEINE HILFE

Falls Sie in diesem Hilfekapitel Informationen vermissen oder Wünsche für Ergänzungen haben, wenden Sie sich per E-Mail an documentation@copadata.com.

PROJEKTUNTERSTÜTZUNG

Unterstützung bei Fragen zu konkreten eigenen Projekten erhalten Sie vom Customer Service, den Sie per E-Mail an support@copadata.com erreichen.

LIZENZEN UND MODULE

Sollten Sie feststellen, dass Sie weitere Module oder Lizenzen benötigen, sind unsere Mitarbeiter unter sales@copadata.com gerne für Sie da.

2 ISPIP

Der **ISPIP Treiber** kommuniziert mit **Hekatron BMZ-Integral (ClientIdentifierClass 0x80)** Brandmeldeanlagen über das ISP-IP Protokoll. Dabei wird die Kommunikation mit Hauptzentralen und Unterzentralen unterstützt.

Funktionalität:

- Manuelle Erstellung und Adressierung der Variablen.
- Statusmeldungen nach CAMP werden in Leserichtung unterstützt.

- Analog-Meldungen nach AVP werden in Leserichtung unterstützt.
- Kommandos für Statusänderungen werden unterstützt.
- Es wird die interne Zeitstempelung verwendet. Externe Zeitstempel werden nicht angefordert.
- Die Zeitsynchronisierung wird nicht unterstützt.

Ablauf:

- Beim Starten der Kommunikation erfolgt eine Generalabfrage für die Statusmeldungen durch den Treiber. Zusätzlich aktiviert der Treiber die Übertragung von Analogwerten.
- Für die spontane Kommunikation der Statusmeldungen sendet die Steuerung nur Meldungen für Melder, die sich nicht in dem Normalzustand befinden. Für Melder die sich in dem Normalzustand befinden, wird keine Meldung empfangen bei der Generalabfrage.
- Für die addressierte Variablen wird der Wert *Idle (0)* mit dem Status *SPONT* angenommen, wenn auf eine Generalabfrage keinen abweichenden Status empfangen wird.
- Die Verbindung wird mit einer automatischen Laufzeitüberwachung (= Watchdog) kontrolliert.
- In der Anlage muss für jede Verbindung ein Fremdleitsystem konfiguriert werden. Für den Redundanzbetrieb müssen somit zwei Fremdleitsysteme konfiguriert sein.

3 ISPIP - Datenblatt

Allgemein:	
Treiberdateiname	ISPIP.exe
Treiberbezeichnung	ISPIP Hekatron BMZ
Steuerungs-Typen	BMZ-Integral von Hekatron. Ankopplung an der Hauptzentralen oder an der Unterzentrale.
Steuerungs-Hersteller	Hekatron

Treiber unterstützt:	
Protokoll	unknown
Adressierung: Adress-basiert	Address based
Adressierung:	--

Treiber unterstützt:	
Namens-basiert	
Kommunikation spontan	X
Kommunikation pollend	--
Online Browsing	--
Offline Browsing	--
Echtzeitfähig	--
Blockwrite	--
Modemfähig	--
RDA numerisch	--
RDA String	--
Hysterese	--
erweiterte API	X
Unterstützung von Statusbit **WR-SUC**	--
alternative IP-Adresse	--

Voraussetzungen:	
Hardware PC	Standard Netzwerkkarte
Software PC	-
Hardware Steuerung	-
Software Steuerung	Ein Fremdsystem mit Benutzername und Passwort muss konfiguriert sein für den Zugriff. Für den Redundanzbetrieb mit Server / Standby Server müssen zwei Fremdsysteme konfiguriert sein.
Benötigt v-dll	X

Plattformen:	
Betriebssysteme	Windows 10; Windows 7; Windows 8; Windows 8.1; Windows Server 2008 R2; Windows Server 2012; Windows Server 2012 R2; Windows Server 2016

4 Treiber-Historie

Datum	Build Nummer	Änderung
22.06.18	48508	Treiberdokumentation wurde neu erstellt

5 Voraussetzungen

Dieses Kapitel enthält Informationen zu den Voraussetzungen, die für die Verwendung des Treibers erforderlich sind.

5.1 PC

Für die Kommunikation des Treibers müssen auf dem Rechner folgende Voraussetzungen erfüllt sein:

- Der Treiber stellt eine ausgehende TCP/IP Verbindung zur SPS über eine Standard-Netzwerkkarte her.
- Benutzername und Passwort (Kennung) für den Zugriff auf die Steuerung müssen bekannt und in der Treiberkonfiguration konfiguriert sein.

5.2 Steuerung

Für die Kommunikation des Treibers müssen auf der SPS folgende Voraussetzungen erfüllt sein:

- Die Steuerung muss für den externen Zugriff eines Fremdsystems mit entsprechendem Benutzernamen und Passwort konfiguriert sein.
- Die Steuerung muss über TCP/IP erreichbar sein.

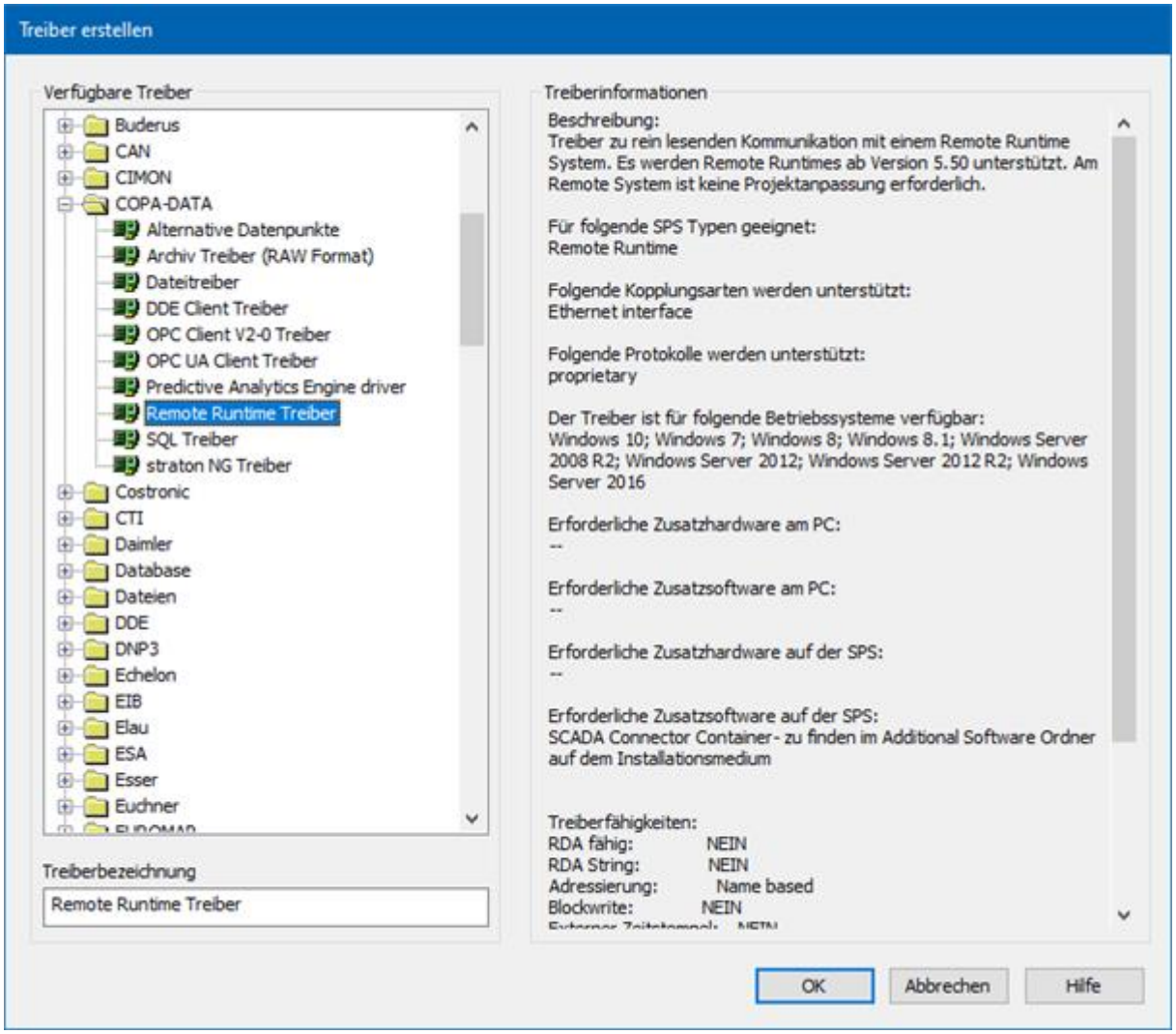
6 Konfiguration

In diesem Kapitel lesen Sie, wie Sie den Treiber im Projekt anlegen und welche Einstellungen beim Treiber möglich sind.

> **Info**
>
> Weitere Einstellungen, die Sie für Variablen in zenon vornehmen können, finden Sie im Kapitel Variablen der Online-Hilfe.

6.1 Anlegen eines Treibers

Im Dialog **Treiber erstellen** wählen Sie aus einer Liste jenen Treiber, den Sie neu anlegen wollen.



Parameter	Beschreibung
Verfügbare Treiber	Liste aller verfügbaren Treiber. Die Darstellung erfolgt in einer Baumstruktur: *[+]* erweitert die Ordnerstruktur und zeigt die darin enthaltenen Treiber. [-] reduziert die Ordnerstruktur Default: *keine Auswahl*
Treiberbezeichnung	Eindeutige **Bezeichnung** des Treibers. Default: *leer* Das Eingabefeld wird nach Auswahl eines Treibers aus der Liste der verfügbaren Treiber mit der vordefinierten **Bezeichnung** vorausgefüllt.
Treiberinformationen	Weiterführende Informationen über den gewählten Treiber. Default: *leer* Nach Auswahl eines Treibers werden in diesem Bereich die Informationen zum gewählten Treiber angezeigt.

DIALOG BEENDEN

Option	Beschreibung
OK	Übernimmt alle Einstellungen und öffnet den Treiberkonfigurationsdialog des ausgewählten Treibers.
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

Info

Die Inhalte dieses Dialogs sind in der Datei Treiber_[Sprachkürzel].xml gespeichert. Sie finden diese Datei im Ordner *C:\ProgramData\COPA-DATA\zenon[Versionsnummer]*.

TREIBER NEU ANLEGEN

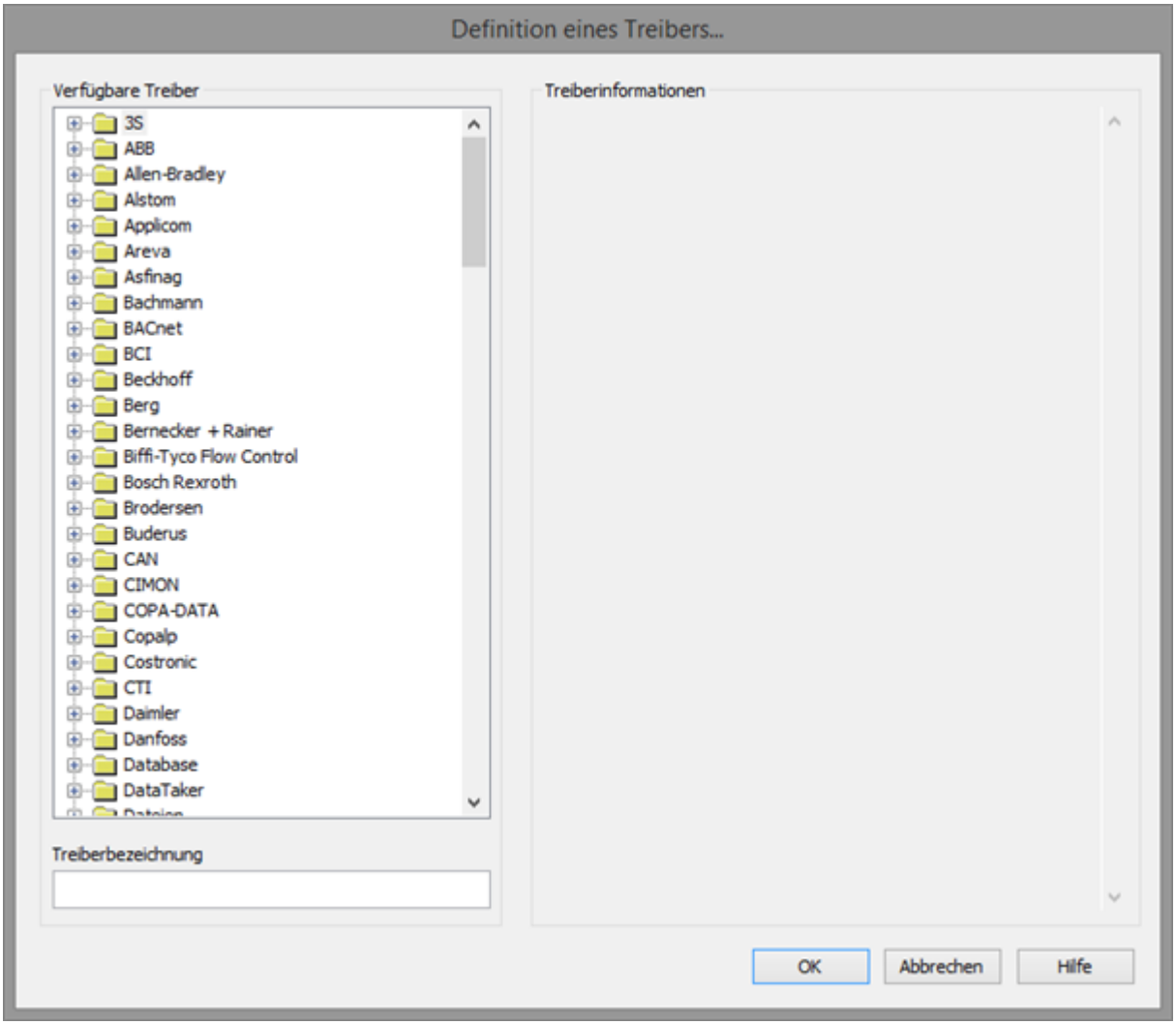
Um einen neuen Treiber anzulegen:

1. Klicken Sie mit der rechten Maustaste im Projektmanager auf **Treiber** und wählen Sie im Kontextmenü **Treiber neu** aus.

   Optional: Wählen Sie die Schaltfläche **Treiber neu** aus der Symbolleiste der Detailansicht der **Variablen**.

   Der Dialog **Treiber erstellen** wird geöffnet.

2. Der Dialog bietet eine Auflistung aller verfügbaren Treiber an.



3. Wählen Sie den gewünschten Treiber und benennen Sie diesen im Eingabefeld **Treiberbezeichnung**.
   Dieses Eingabefeld entspricht der Eigenschaft **Bezeichnung**. Per Default wird der Name des ausgewählten Treibers in diesem Eingabefeld automatisch eingefügt.
   Für die **Treiberbezeichnung** gilt:

   - Die **Treiberbezeichnung** muss eindeutig sein.
     Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.

Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.

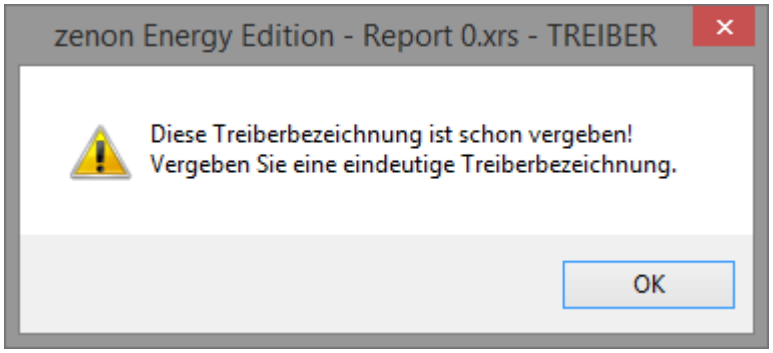
- Die **Treiberbezeichnung** ist Bestandteil des Dateinamens.
  Daher darf Sie nur Zeichen enthalten, die vom Betriebssystem unterstützt werden. Nicht gültige Zeichen werden durch einen Unterstrich (_) ersetzt.
- **Achtung:** Die Bezeichnung kann später nicht mehr geändert werden.

4. Bestätigen Sie den Dialog mit Klick auf die Schaltfläche **OK**.
   Der Konfigurationsdialog des ausgewählten Treibers wird geöffnet.

Hinweis: Treibernamen sind nicht sprachumschaltbar. Sie werden später immer in der Sprache angezeigt, in der sie angelegt wurden, unabhängig von der Sprache des Editors. Das gilt auch für Treiberobjekttypen.

DIALOG TREIBERBEZEICHNUNG BEREITS VORHANDEN

Ist ein Treiber bereits im Projekt vorhanden wird dies in einem Dialog angezeigt. Mit Klick auf die Schaltfläche **OK** wird der Warndialog geschlossen. Der Treiber kann korrekt benannt werden.



ZENON PROJEKT

Bei neu angelegten Projekten werden die folgenden Treiber automatisch angelegt:

- **Intern**
- **MathDr32**
- **SysDrv**

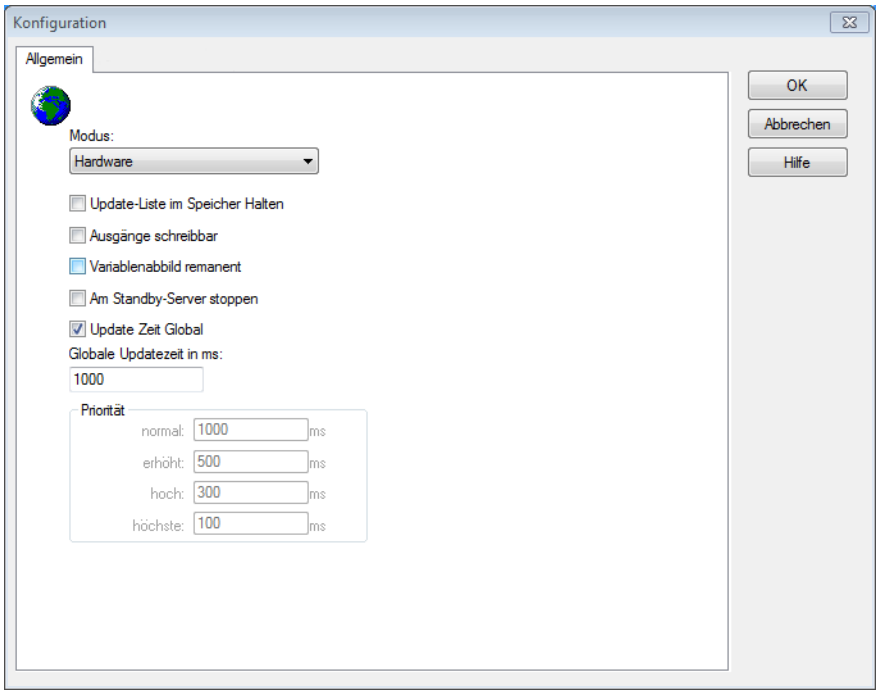
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein.
> Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

6.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

6.2.1 Allgemein

Beim Anlegen eines Treibers wird der Konfigurationsdialog geöffnet. Um den Dialog später zum Bearbeiten zu öffnen, führen Sie einen Doppelklick auf den Treiber in der Liste aus oder klicken Sie auf die Eigenschaft **Konfiguration**.



Option	Beschreibung
Modus	Ermöglicht ein Umschalten zwischen Hardware und Simulationsmodus ▸ *Hardware*: Die Verbindung zur Steuerung wird hergestellt. ▸ Simulation - statisch: Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus bleiben die Werte konstant oder die Variablen behalten die über zenon Logic gesetzen Werte. Jede Variable hat seinen eigenen Speicherbereich. Zum Beispiel zwei Variablen vom Typ Merker mit Offset 79, können zur Runtime unterschiedliche Werte haben und beeinflussen sich gegenseitig nicht. Ausnahme: Der

Option	Beschreibung
	Simulatortreiber. - *Simulation - zählend*: Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus zählt der Treiber die Werte innerhalb ihres Wertebereichs automatisch hoch. - *Simulation - programmiert*: Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in einer in den Treiber integrierten zenon Logic Runtime ab. Details siehe Kapitel Treibersimulation.
Update-Liste im Speicher Halten	Einmal angeforderte Variablen werden weiterhin von der Steuerung angefordert, auch wenn diese aktuell nicht mehr benötigt werden. Dies hat den Vorteil, dass z B. mehrmalige Bildumschaltungen nach dem erstmaligen Aufschalten beschleunigt werden, da die Variablen nicht neu angefordert werden müssen. Der Nachteil ist eine erhöhte Belastung der Kommunikation zur Steuerung.
Ausgänge schreibbar	- *Aktiv*: Ausgänge können beschrieben werden. - *Inaktiv*: Das Beschreiben der Ausgänge wird unterbunden. **Hinweis**: Steht nicht für jeden Treiber zur Verfügungen.
Variablenabbild remanent	Diese Option speichert und restauriert den aktuellen Wert, den Zeitstempel und die Status eines Datenpunkts. Grundvoraussetzung: Die Variable muss einen gültigen Wert und Zeitstempel besitzen. Das Variablenabbild wird im Modus Hardware gespeichert, wenn einer dieser Status aktiv ist: - Benutzerstatus *M1* (*0*) bis *M8* (*7*) - *REVISION(9)* - *AUS(20)*

Option	Beschreibung
	▸ *ERSATZWERT(27)* Das Variablenabbild wird immer gespeichert wenn: ▸ die Variable vom Objekttyp **Kommunikationsdetails** ist ▸ der Treiber im Simulationsmodus läuft. (nicht programmierte Simulation) Folgende Status werden beim Start der Runtime nicht restauriert: ▸ *SELECT(8)* ▸ *WR-ACK(40)* ▸ *WR-SUC(41)* Der Modus **Simulation - programmiert** beim Treiberstart ist kein Kriterium, um das remanente Variablenabbild zu restaurieren.
Am Standby Server stoppen	Einstellung für Redundanz bei Treibern, die nur eine Kommunikationsverbindung erlauben. Dazu wird der Treiber am Standby Server gestoppt und erst beim Hochstufen wieder gestartet. **Achtung:** Ist diese Option aktiv, ist die lückenlose Archivierung nicht mehr gewährleistet. ▸ *Aktiv*: Versetzt den Treiber am nicht-prozessführenden Server automatisch in einen Stopp-ähnlichen Zustand. Im Unterschied zum Stoppen über Treiberkommando erhält die Variable nicht den Status **abgeschaltet**, sondern einen leeren Wert. Damit wird verhindert, dass beim Hochstufen zum Server nicht relevante Werte in AML , CEL und Archiv erzeugt werden. Default: *inaktiv* **Hinweis:** Nicht verfügbar, wenn CE Terminal als Datenserver dient. Weitere Informationen dazu erhalten Sie im Handbuch zenon Operator im Kapitel CE Terminal als Datenserver.
Update Zeit Global	Einstellung für globale Update-Zeiten in Millisekunden:

Option	Beschreibung
	▸ *Aktiv*: Die eingestellte **Globale Update Zeit** wird für alle Variablen im Projekt verwendet. Die bei den Variablen eingestellte Priorität wird nicht verwendet. ▸ *Inaktiv*: Die eingestellten Prioritäten werden für die einzelnen Variablen verwendet. **Ausnahmen:** Spontane Treiber ignorieren diese Option. Sie nutzen in der Regel die kürzest mögliche Update Zeit. Details siehe Abschnitt **Update Zeit spontane Treiber**.
Priorität	Hier werden die Pollingzeiten der einzelnen Prioritätsklassen eingestellt. Alle Variablen mit der entsprechenden Priorität werden in der eingestellten Zeit gepollt. Die Zuordnung der Variablen erfolgt separat bei jeder Variablen über die Einstellungen in den Variableneigenschaften. Mit den Prioritätsklassen kann die Kommunikation der einzelnen Variablen auf die Wichtigkeit oder benötigte Aktualität abgestuft werden. Daraus ergibt sich eine verbesserte Verteilung der Kommunikationslast. **Achtung:** Prioritätsklassen werden nicht von jedem Treiber unterstützt, z.B. von spontan kommunizierenden zenon Treibern.

DIALOG BEENDEN

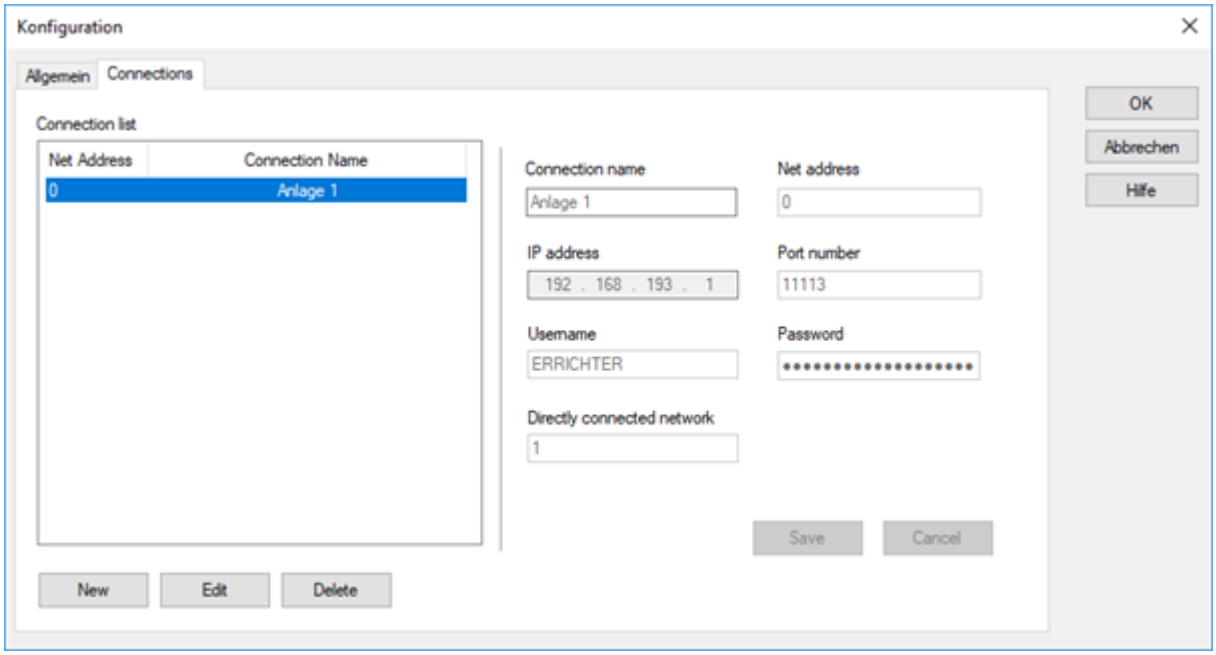
Option	Beschreibung
OK	Übernimmt alle Änderungen in allen Registerkarten und schließt den Dialog.
Abbrechen	Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

UPDATE ZEIT SPONTANE TREIBER

Bei spontanen Treibern wird beim **Sollwert Setzen**, **Advisen** von Variablen und bei **Requests** sofort ein Lesezyklus ausgelöst - unabhängig von der eingestellten Update Zeit. Damit wird sicher gestellt, dass der Wert nach dem Schreiben in der Visualisierung sofort zur Verfügung steht. In der Regel beträgt die Updatezeit 100 ms.

Spontane Treiber sind **ArchDrv**, **BiffiDCM**, **BrTcp32**, **DNP3**, **Esser32**, **FipDrv32**, **FpcDrv32**, **IEC850**, **IEC870**, **IEC870_103**, **Otis**, **RTK9000**, **S7DCOS**, **SAIA_Slave**, **STRATON32** und **Trend32**.

6.2.2 Connections



Parameter	Beschreibung
Connection name	Frei konfigurierbarer Namen für die Steuerung
New	Legt eine neue Verbindung mit Default-Einstellungen an. Die Eigenschaften der Verbindung werden für die Projektierung freigeschalten.
Edit	Öffnet die ausgewählte bestehende Verbindung zur Bearbeitung. Die Eigenschaften der Verbindung werden für die Projektierung freigeschalten.
Delete	Die ausgewählte Verbindung wird ohne Rückfrage gelöscht.

Parameter	Beschreibung
Net address	Eindeutige Netzadresse der SPS im Netzwerk. Die Netzaddresse identifiziert die Verbindung. Jede Verbindung benötigt eine eindeutige Netzaddresse. Die hier projektierte Netzaddresse muss auch bei jenen Variablen, welche diese Verbindung nutzen wollen, entsprechend konfiguriert sein. Für die Kommunikation zur SPS ist diese Netzaddresse nicht erforderlich.
IP adress	IPv4 Addresse der Steuerung.
Port Number	Nummer des TCP-Ports der Steuerung. Default: *11113*
Benutzername	Benutzername für den Benutzer, welcher in der Steuerung für den Zugriff des Fremdsystems konfiguriert wurde. **Hinweis:** Bei der Projektierung muss die Groß-/Kleinschreibung berücksichtigt werden.
Passwort	Passwort für den Benutzer, welches in der Steuerung für den Zugriff des Fremdsystems konfiguriert wurde. Groß-/Kleinschreibung muss berücksichtigt werden.
Directly Connected Network	Die Addresse der Zentrale mit dem die Verbindung hergestellt wird.
Save	Speichert eine neue oder bearbeitete Verbindung.
Cancel	Beendet das Bearbeiten der Verbindungseinstellungen. Geänderte Projektierungen der zu bearbeitenden Verbindung werden nicht gespeichert.

DIALOG BEENDEN

Option	Beschreibung
OK	Übernimmt alle Änderungen in allen Registerkarten und schließt den Dialog.

Option	Beschreibung
Abbrechen	Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

7 Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

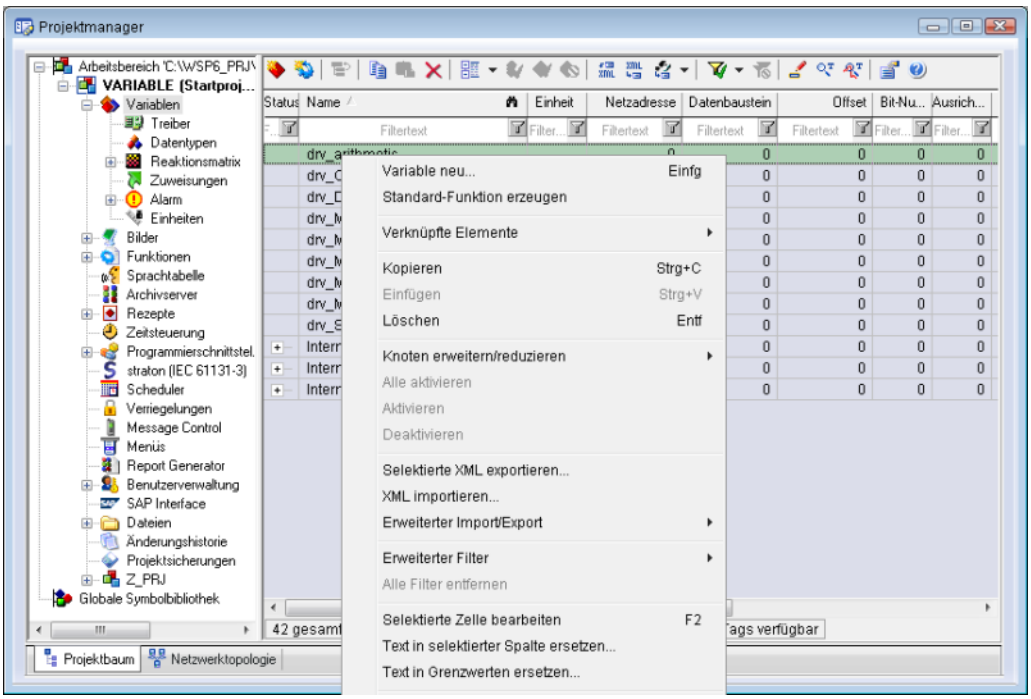
Variablen können angelegt werden:

- als einfache Variable
- in Arrays
- als Struktur-Variablen

DIALOG VARIABLE

Um eine neue Variable zu erstellen, gleich welchen Typs:

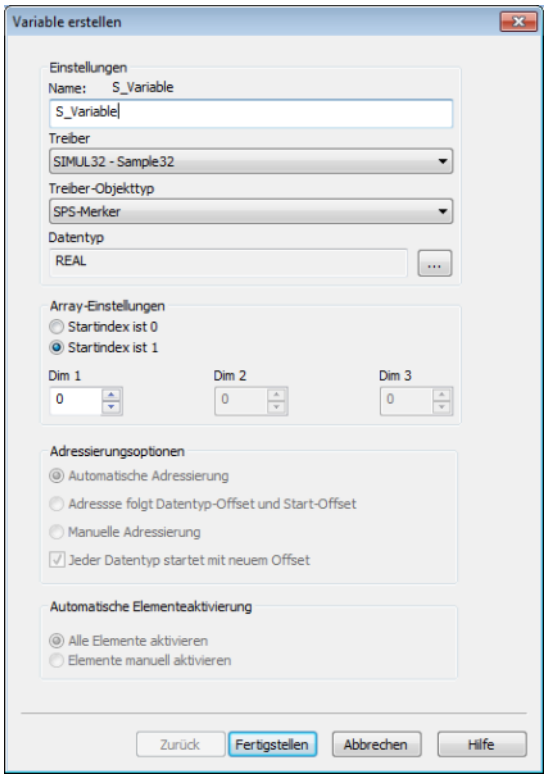
1. Wählen Sie im Knoten **Variablen** im Kontextmenü den Befehl **Variable neu**.



Der Dialog zur Konfiguration der Variable wird geöffnet.

2. Konfigurieren Sie die Variable.
3. Welche Einstellungen möglich sind, hängt ab vom Typ der Variablen.

DIALOG VARIABLE ERSTELLEN



Eigenschaft	Beschreibung
Name	Eindeutiger Name der Variablen. Ist eine Variable mit gleichem Namen im Projekt bereits vorhanden, kann keine weitere Variable mit diesem Namen angelegt werden. Maximale Länge: 128 Zeichen **Achtung:** Die Zeichen **#** und **@** sind für Variablennamen nicht erlaubt. Bei Verwendung nicht zugelassener Zeichen kann die Variablenerstellung nicht abgeschlossen werden, die Schaltfläche **Fertigstellen** bleibt inaktiv. **Hinweis:** Manche Treiber erlauben die Adressierung auch über die Eigenschaft **Symbolische Adresse**.
Treiber	Wählen Sie aus der Dropdownliste den gewünschten Treiber. **Hinweis:** Sollte im Projekt noch kein Treiber angelegt sein, wird automatisch der Treiber für interne Variable (**Intern.exe**) geladen.
Treiberobjekttyp	Wählen Sie aus der Dropdownliste den passenden Treiberobjekttyp aus.
Datentyp	Wählen Sie den gewünschten Datentyp. Klick auf die Schaltfläche

Eigenschaft	Beschreibung
	... öffnet den Auswahl-Dialog.
Array-Einstellungen	Erweiterte Einstellungen für Array-Variablen. Details dazu lesen Sie im Abschnitt Arrays.
Adressierungsoptionen	Erweiterte Einstellungen für Arrays und Struktur-Variablen. Details dazu lesen Sie im jeweiligen Abschnitt.
Automatische Elementeaktivierung	Erweiterte Einstellungen für Arrays und Struktur-Variablen. Details dazu lesen Sie im jeweiligen Abschnitt.

SYMBOLISCHE ADRESSE

Die Eigenschaft **Symbolische Adresse** kann für die Adressierung alternativ zu **Name** oder **Kennung** der Variablen verwendet werden. Die Auswahl erfolgt im Treiberdialog, die Konfiguration in der Variableneigenschaft. Beim Import von Variablen unterstützter Treiber wird die Eigenschaft automatisch eingetragen.

Maximale Länge: *1024* Zeichen.

Folgende Treiber unterstützen die **Symbolische Adresse**:

- **3S_V3**
- **AzureDrv**
- **BACnetNG**
- **IEC850**
- **KabaDPServer**
- **OPCUA32**
- **Phoenix32**
- **POZYTON**
- **RemoteRT**
- **S7TIA**
- **SEL**
- **SnmpNg32**
- **PA_Drv**
- **EUROMAP63**

ABLEITUNG VOM DATENTYP

Messbereich, **Signalbereich** und **Sollwert Setzen** werden immer:

- vom Datentyp abgeleitet
- beim Ändern des Datentyps automatisch angepasst

Hinweis Signalbereich: Bei einem Wechsel auf einen Datentyp, der den eingestellten **Signalbereich** nicht unterstützt, wird der **Signalbereich** automatisch angepasst. Zum Beispiel wird bei einem Wechsel von **INT** auf **SINT** der **Signalbereich** auf *127* geändert. Die Anpassung erfolgt auch dann, wenn der **Signalbereich** nicht vom Datentyp abgeleitet wurde. In diesem Fall muss der **Messbereich** manuell angepasst werden.

7.2 Adressierung

Variablen für den **ISPIP Treiber** müssen manuell erstellt und addressiert werden.

Gruppe/Eigenschaft	Beschreibung
Allgemein	
Name	Frei vergebbarer Name. Achtung: Je zenon Projekt muss der Name eindeutig sein.
Kennung	Frei vergebbare Kennung, z. B. für Betriebsmittelkennung, Kommentar...
Adressierung	
Netzadresse	Busadresse oder Netzadresse der Variablen. Diese Adresse bezieht sich auf die Busadresse der Verbindungsprojektierung im Treiber. Damit wird ausgewählt auf welcher Steuerung sich die Variable befindet.
Datenbaustein	Wird für diesen Treiber nicht verwendet.
Offset	Wird für diesen Treiber nicht verwendet.
Ausrichtung	Wird für diesen Treiber nicht verwendet.
Bitnummer	Wird für diesen Treiber nicht verwendet.
Stringlänge	Wird für diesen Treiber nicht verwendet.
Treiber Anbindung/Treiber objekttyp	Objekttyp der Variablen. Wird abhängig vom verwendeten Treiber beim Erstellen der Variablen ausgewählt und kann hier geändert werden.
Treiber Anbindung/Datentyp	Datentyp der Variablen. Wird beim Erstellen der Variablen ausgewählt und kann hier geändert werden. **Achtung:** Wird der Datentyp nachträglich geändert, müssen alle

Gruppe/Eigenschaft	Beschreibung
	anderen Eigenschaften der Variable überprüft und angepasst werden.
Network number	Die Nummer der Hauptzentrale oder Teilzentrale.
Element type	Auswahl des Hauptelement-Typs. Auswahl aus Dropdownliste.
Element subtype	Auswahl des Subelementtyps aus Dropdownliste. Für Variablen vom Treiberobjekttyp *State*. Die Auswahl via Dropdownliste ist abhängig vom gewählten Hauptelement-Typ
Element number	Die Elementnummer im Netzwerk.
Element subnumber	Die Subelement-Nummer des Elements.
AV type	Für Variablen vom Treiberobjekttyp *Analog Value*. Auswahl des Analogwertes aus Dropdownliste. Beachten Sie: Nicht alle Melder unterstützen alle Analogwerte. Der Hauptelementtyp (Variableneigenschaft **Element type AV**) ist fixiert auf "*Zone*" (Meldergruppe).
Function type	Für Variablen vom Treiberobjekttyp *Command*. Beachten Sie: Nicht alle Elemente unterstützen alle Kommandotypen.

7.3 Treiberobjekte und Datentypen

Treiberobjekte sind in der Steuerung verfügbare Bereiche wie z. B. Merker, Datenbausteine usw. Hier lesen Sie, welche Treiberobjekte vom Treiber zur Verfügung gestellt werden und welche IEC-Datentypen dem jeweiligen Treiberobjekt zugeordnet werden können.

7.3.1 Treiberobjekte

Folgende Objekttypen stehen in diesem Treiber zur Verfügung:

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Beschreibung
Status	66	X	--	*WORD*	Treiberobjekttyp für Zustandswerte. ▸ Die hochwertigen 8 Bits stellen den Hauptstatus des Elementes dar. ▸ Die niederwertigen 8 Bits stellen den Substatus des Elementes dar. Der Wert wird zur Laufzeit mit dem Wert 0 und dem Status SPONT initialisiert . Der Treiber bekommt von der SPS (BMZ-Integral) in der Antwort auf eine Summenabfrage, nur jene Zustandswerte kommuniziert, die vom Normalzustand abweichen.
Analog Value	65	X	--	*SINT*	Treiberobjekttyp für Analogwerte. Die Übertragung von Analogwerten erfolgt spontan. Nach Start des Treibers wird die Übertragung von Analogwerten aktiviert.
Command	67	X	*X*	*WORD*	Treiberobjekttyp für Kommandos. Abhängig vom Typ des Kommandos wird die erfolgreiche Ausführung von der BMZ-Integral bestätigt: ▸ durch eine Antwort

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Beschreibung
					auf das Kommando oder ▸ mittels Statusänderung des Elements Nach erfolgreicher Kommunikation dieser Statusänderung wird das Kommando ausgeführt.
Kommunikationsdetails	35	X	*X*	*BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING*	Variablen für die statische Analyse der Kommunikation; wird zwischen Treiber und Runtime übertragen (nicht zur SPS). **Hinweis**: Die Adressierung und das Verhalten ist bei den meisten zenon Treibern gleich Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 35).

Legende:

X: wird unterstützt

--: wird nicht unterstützt

KANALTYP

Der Begriff "**Kanaltyp**" ist die interne numerische Bezeichnung des Treiberobjekttyps. Diese wird auch für den erweiterten DBF Import/Export der Variablen verwendet.

"**Kanaltyp**" wird für den erweiterten CSV Import/Export der Variablen in der Spalte "**HWObjectType**" verwendet.

7.3.2 Zuordnung der Datentypen

Alle Variablen in zenon werden von IEC-Datentypen abgeleitet. In folgender Tabelle werden zur besseren Übersicht die IEC-Datentypen den Datentypen der Steuerung gegenübergestellt.

Steuerung	zenon	Datenart
-	BOOL	8
-	USINT	9
Analogwerte	SINT	10
-	UINT	2
-	INT	1
-	UDINT	4
-	DINT	3
-	ULINT	27
-	LINT	26
-	REAL	5
-	LREAL	6
-	STRING	12
-	WSTRING	21
-	DATE	18
-	TIME	17
-	DATE_AND_TIME	20
-	TOD (Time of Day)	19
Statusmeldungen, Kommandos	WORD	23

DATENART

Der Begriff **Datenart** ist die interne numerische Bezeichnung des Datentyps. Diese wird auch für den erweiterten DBF Import/Export der Variablen verwendet.

7.4 Variablen anlegen durch Import

Variablen können auch mittels Variablenimport angelegt werden. Für jeden Treiber stehen XML- und DBF-Import zur Verfügung.

> **Info**
>
> Details zu Import und Export von Variablen finden Sie im Handbuch Import-Export im Abschnitt Variablen.

7.4.1 XML Import

Beim XML- Import von Variablen oder Datentypen werden diese erst einem Treiber zugeordnet und dann analysiert. Vor dem Import entscheidet der Benutzer, ob und wie das jeweilige Element (Variable oder Datentyp) importiert werden soll:

- *Importieren*:
  Das Element wird neu importiert.
- *Überschreiben*:
  Das Element wird importiert und überschreibt ein bereits vorhandenes Element.
- *Nicht importieren*:
  Das Element wird nicht importiert.

Hinweis: Beim Import werden die Aktionen und deren Dauer in einem Fortschrittsbalken angezeigt. In der folgenden Dokumentation wird der Import von Variablen beschrieben. Datentypen werden analog dazu importiert.

VORAUSSETZUNGEN

Beim Import gelten folgende Bedingungen:

- **Abwärtskompatibilität**

  Beim XML Import/Export ist keine Abwärtskompatibilität gegeben. Daten aus älteren zenon Versionen können übernommen werden. Die Übergabe von Daten aus neueren Versionen an ältere wird nicht unterstützt.

- **Konsistenz**

  Die zu importierende XML-Datei muss konsistent sein. Beim Import der Datei erfolgt keine Plausibilitätsprüfung. Weisen die importierten Daten Fehler auf, kann es zu unerwünschten Effekten im Projekt kommen.

  Dies muss vor allem auch beachtet werden, wenn in einer XML-Datei nicht alle Eigenschaften vorhanden sind und diese dann durch Default-Werte ersetzt werden. Z. B.: Eine binäre Variable hat einen Grenzwert von *300*.

- **Struktur-Datentypen**

  Struktur-Datentypen müssen über die gleiche Anzahl von Strukturelementen verfügen. Beispiel: Ein Strukturdatentyp im Projekt hat 3 Strukturelemente. Ein gleichnamiger Datentyp in der XML-Datei hat 4 Strukturelemente. Dann wird keine der auf diesem Datentyp basierenden Variablen der Datei in das Projekt importiert.

Tipp

Weitere Informationen zum XML-Import finden Sie im Handbuch **Import - Export**, im Kapitel **XML-Import**.

7.4.2 DBF Import/Export

Daten können nach dBase exportiert und aus dBase importiert werden.

Info

Import und Export über CSV oder dBase unterstützt keine treiberspezifischen Variableneinstellungen wie z. B. Formeln. Nutzen Sie dafür den Export/Import über XML.

IMPORT DBF-DATEI

Um den Import zu starten:

1. Führen Sie einen Rechtsklick auf die Variablenliste aus.
2. Wählen Sie in der Dropdownliste von **Erweiterter Export/Import** ... den Befehl **dBase importieren**.
3. Folgen Sie den Anweisungen des Importassistenten.

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

> **Info**
>
> Beachten Sie:
>
> - Treiberobjekttyp und Datentyp müssen in der DBF-Datei an den Zieltreiber angepasst werden, damit Variablen importiert werden.
> - dBase unterstützt beim Import keine Strukturen oder Arrays (komplexe Variablen).

EXPORT DBF-DATEI

Um den Export zu starten:

1. Führen Sie einen Rechtsklick auf die Variablenliste aus.
2. Wählen Sie im Dropdownliste von **Erweiterter Export/Import** ... den Befehl **dBase exportieren...** .
3. Folgen Sie den Anweisungen des Exportassistenten.

> **Achtung**
>
> DBF-Dateien:
>
> - müssen in der Benennung dem 8.3 DOS Format für Dateinamen entsprechen (8 alphanumerische Zeichen für Name, 3 Zeichen Erweiterung, keine Leerzeichen)
> - dürfen im Pfadnamen keinen Punkt (.) enthalten.
> Z. B. ist der Pfad *C:\users\Max.Mustermann\test.dbf* ungültig.
> Gültig wäre: *C:\users\MaxMustermann\test.dbf*
> - müssen nahe am Stammverzeichnis (Root) abgelegt werden, um die eventuelle Beschränkungen für Dateinamenlänge inklusive Pfad zu erfüllen: maximal 255 Zeichen

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

> **Info**
>
> dBase unterstützt beim Export keine Strukturen oder Arrays (komplexe Variablen).

DATEIAUFBAU DER DBASE EXPORTDATEI

Für den Variablenimport und -export muss die dBaseIV-Datei folgende Struktur und Inhalte besitzen.

Achtung

dBase unterstützt keine Strukturen oder Arrays (komplexe Variablen).

DBF-Dateien müssen:

- in der Benennung dem 8.3 DOS Format für Dateinamen entsprechen (8 alphanumerische Zeichen für Name, 3 Zeichen Erweiterung, keine Leerzeichen)
- nahe am Stammverzeichnis (Root) abgelegt werden

STRUKTUR

Bezeichnung	Typ	Feldgröße	Bemerkung
KANALNAME	Char	128	Variablenname. Länge kann über den Eintrag **MAX_LAENGE** in der **project.ini** eingeschränkt werden.
KANAL_R	C	128	Ursprünglicher Name einer Variablen, der durch den Eintrag unter VARIABLENNAME ersetzt werden soll (Feld/Spalte muss manuell angelegt werden). Länge kann über den Eintrag **MAX_LAENGE** in der **project.ini** eingeschränkt werden.
KANAL_D	Log	1	Variable wird bei Eintrag *1* gelöscht (Feld/Spalte muss manuell angelegt werden).
TAGNR	C	128	Kennung. Länge kann über den Eintrag **MAX_LAENGE** in der **project.ini** eingeschränkt werden.
EINHEIT	C	11	Technische Maßeinheit
DATENART	C	3	Datentyp (z. B. Bit, Byte, Wort, ...) entspricht dem Datentyp.
KANALTYP	C	3	Speicherbereich in der SPS (z. B. Merkerbereich, Datenbereich, ...) entspricht Treiberobjekttyp.
HWKANAL	Num	3	Netzadresse
BAUSTEIN	N	3	Datenbaustein-Adresse (nur bei Variablen aus den Datenbereich der SPS)

Bezeichnung	Typ	Feldgröße	Bemerkung
ADRESSE	N	5	Offset
BITADR	N	2	Für Bit-Variablen: Bitadresse Für Byte-Variablen: 0=niederwertig, 8=höherwertig Für String-Variablen: Stringlänge (max. 63 Zeichen)
ARRAYSIZE	N	16	Anzahl der Variablen im Array für Index-Variablen ACHTUNG: Nur die erste Variable steht voll zur Verfügung. Alle folgenden sind nur über VBA oder den Rezeptgruppen Manager zugänglich
LES_SCHR	L	1	Lese-Schreib-Berechtigung 0: Sollwert setzen ist nicht erlaubt 1: Sollwert setzen ist erlaubt
MIT_ZEIT	L	1	Zeitstempelung in zenon (nur wenn vom Treiber unterstützt)
OBJEKT	N	2	Treiberspezifische ID-Nummer des Primitivobjekts setzt sich zusammen aus TREIBER-OBJEKTTYP und DATENTYP
SIGMIN	Float	16	Rohwertsignal minimal (Signalauflösung)
SIGMAX	F	16	Rohwertsignal maximal (Signalauflösung)
ANZMIN	F	16	technischer Wert minimal (Messbereich)
ANZMAX	F	16	technischer Wert maximal (Messbereich)
ANZKOMMA	N	1	Anzahl der Nachkommastellen für die Darstellung der Werte (Messbereich)
UPDATERATE	F	19	Updaterate für Mathematikvariablen (in sec, eine Dezimalstelle möglich) bei allen anderen Variablen nicht verwendet
MEMTIEFE	N	7	Nur aus Kompatibilitätsgründen vorhanden
HDRATE	F	19	HD-Updaterate für hist. Werte (in sec, eine Dezimalstelle möglich)
HDTIEFE	N	7	HD-Eintragtiefe für hist. Werte (Anzahl)
NACHSORT	L	1	HD-Werte als nachsortierte Werte
DRRATE	F	19	Aktualisierung an die Ausgabe (für zenon DDE-Server,

Bezeichnung	Typ	Feldgröße	Bemerkung
			in sec, eine Kommastelle möglich)
HYST_PLUS	F	16	Positive Hysterese; ausgehend vom Messbereich
HYST_MINUS	F	16	Negative Hyterese; ausgehend vom Messbereich
PRIOR	N	16	Priorität der Variable
REAMATRIZE	C	32	Name der zugeordnete Reaktionsmatrix
ERSATZWERT	F	16	Ersatzwert; ausgehend vom Messbereich
SOLLMIN	F	16	Sollwertgrenze Minimum; ausgehend vom Messbereich
SOLLMAX	F	16	Sollwertgrenze Maximum; ausgehend vom Messbereich
VOMSTANDBY	L	1	Variable vom Standby Server anfordern; der Wert der Variable wird im redundanten Netzwerkbetrieb nicht vom Server sondern vom Standby Server angefordert
RESOURCE	C	128	Betriebsmittelkennung. Freier String für Export und Anzeige in Listen. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
ADJWVBA	L	1	Nichtlineare Wertanpassung: *0*: Nichtlineare Wertanpassung wird verwendet *1*: Nichtlineare Wertanpassung wird nicht verwendet
ADJZENON	C	128	Verknüpftes VBA-Makro zum Lesen der Variablenwerte für die nichtlineare Wertanpassung.
ADJWVBA	C	128	Verknüpftes VBA-Makro zum Schreiben der Variablenwerte für die nichtlineare Wertanpassung.
ZWREMA	N	16	Verknüpfte Zählwert-Rema.
MAXGRAD	N	16	Maximaler Gradient für die Zählwert-Rema.

Achtung

Beim Import müssen Treiberobjekttyp und Datentyp in der DBF-Datei an den Zieltreiber angepasst werden, damit Variablen importiert werden.

GRENZWERTDEFINITION

Grenzwertdefinition für Grenzwert *1* bis *4*, oder Zustand *1* bis *4*:

Bezeichnung	Typ	Feldgröße	Bemerkung
AKTIV1	L	1	Grenzwert aktiv (pro Grenzwert vorhanden)
GRENZWERT1	F	20	technischer Wert oder ID-Nummer der verknüpften Variable für einen dynamischen Grenzwert (siehe VARIABLEx) (wenn unter VARIABLEx *1* steht und hier *-1*, wird die bestehende Variablenzuordnung nicht überschrieben)
SCHWWERT1	F	16	Schwellwert für den Grenzwert
HYSTERESE1	F	14	wird nicht verwendet
BLINKEN1	L	1	Blinkattribut setzen
BTB1	L	1	Protokollierung in CEL
ALARM1	L	1	Alarm
DRUCKEN1	L	1	Druckerausgabe (bei CEL oder Alarm)
QUITTIER1	L	1	quittierpflichtig
LOESCHE1	L	1	löschpflichtig
VARIABLE1	L	1	dyn. Grenzwertverknüpfung der Grenzwert wird nicht durch einen absoluten Wert (siehe Feld GRENZWERTx) festgelegt.
FUNC1	L	1	Funktionsverknüpfung
ASK_FUNC1	L	1	Ausführung über die Alarmmeldeliste
FUNC_NR1	N	10	ID-Nummer der verknüpften Funktion (steht hier -1, so wird die bestehende Funktion beim Import nicht überschrieben)
A_GRUPPE1	N	10	Alarm/Ereignis-Gruppe
A_KLASSE1	N	10	Alarm/Ereignis-Klasse
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Farbe als Windowskodierung

Bezeichnung	Typ	Feldgröße	Bemerkung
GRENZTXT1	C	66	Grenzwerttext
A_DELAY1	N	10	Zeitverzögerung
INVISIBLE1	L	1	Unsichtbar

Bezeichnungen in der Spalte Bemerkung beziehen sich auf die in den Dialogboxen zur Definition von Variablen verwendeten Begriffe. Bei Unklarheiten, siehe Kapitel Variablendefinition.

7.5 Kommunikationsdetails (Treibervariablen)

Das Treiberkit implementiert eine Reihe von Treibervariablen, welche in dem Treiberobjekttyp *Kommunikationsdetails* zusammengefasst sind. Diese sind unterteilt in:

- Information
- Konfiguration
- Statistik und
- Fehlermeldungen

Die Definitionen der im Treiberkit implementierten Variablen sind in der Importdatei **DRVVAR.DBF** verfügbar und können von dort importiert werden.
Pfad zur Datei: *%ProgramData%\COPA-DATA\zenon<Versionsnummer>\PredefinedVariables*

Hinweis: Variablennamen müssen in zenon einzigartig sein. Soll nach einem Import der Variablen vom Treiberobjekttyp *Kommunikationsdetails* aus **DRVVAR.DBF** ein erneuter Import durchgeführt werden, müssen die zuvor importierten Variablen umbenannt werden.

Info

Nicht jeder Treiber unterstützt alle Treibervariablen des Treiberobjekttyps *Kommunikationsdetails*.

Zum Beispiel:

- Variablen für Modem-Informationen werden nur von modemfähigen Treibern unterstützt.
- Treibervariablen für den Polling-Zyklus stehen nur für rein pollende Treiber zur Verfügung.
- Verbindungsbezogene Informationen wie **ErrorMSG** werden nur von Treibern unterstützt, die zu einem Zeitpunkt nur eine Verbindung bearbeiten.

INFORMATION

Name aus Import	Typ	Offset	Erklärung
MainVersion	*UINT*	0	Haupt-Versionsnummer des Treibers.
SubVersion	*UINT*	1	Sub-Versionsnummer des Treibers.
BuildVersion	*UINT*	29	Build-Versionsnummer des Treibers.
RTMajor	*UINT*	49	zenon Hauptversionsnummer
RTMinor	*UINT*	50	zenon Sub-Versionsnummer
RTSp	*UINT*	51	zenon Service Pack-Nummer
RTBuild	*UINT*	52	zenon Buildnummer
LineStateIdle	*BOOL*	24.0	TRUE, wenn die Modemleitung belegt ist.
LineStateOffering	*BOOL*	24.1	TRUE, wenn ein Anruf rein kommt.
LineStateAccepted	*BOOL*	24.2	Der Anruf wird angenommen.
LineStateDialtone	*BOOL*	24.3	Rufton wurde erkannt.
LineStateDialing	*BOOL*	24.4	Wahl aktiv.
LineStateRingBack	*BOOL*	24.5	Während Verbindungsaufbau.
LineStateBusy	*BOOL*	24.6	Zielstation besetzt.
LineStateSpecialInfo	*BOOL*	24.7	Spezielle Statusinformation empfangen.
LineStateConnected	*BOOL*	24.8	Verbindung hergestellt.
LineStateProceeding	*BOOL*	24.9	Wahl ausgeführt.
LineStateOnHold	*BOOL*	24.10	Verbindung in Halten.
LineStateConferenced	*BOOL*	24.11	Verbindung im Konferenzmodus.
LineStateOnHoldPendConf	*BOOL*	24.12	Verbindung in Halten für Konferenz.
LineStateOnHoldPendTransfer	*BOOL*	24.13	Verbindung in Halten für Transfer.
LineStateDisconnected	*BOOL*	24.14	Verbindung beendet.
LineStateUnknow	*BOOL*	24.15	Verbindungszustand nicht bekannt.
ModemStatus	*UDINT*	24	Aktueller Modemstatus.

Name aus Import	Typ	Offset	Erklärung
TreiberStop	*BOOL*	28	Treiber gestoppt Bei *Treiberstop*, hat die Variable den Wert *TRUE* und ein **OFF**-Bit. Nach dem Treiberstart, hat die Variable den Wert *FALSE* und kein **OFF**-Bit.
SimulRTState	*UDINT*	60	Informiert über Status der Runtime bei Treibersimulation.
ConnectionStates	*STRING*	61	Interner Verbindungsstatus des Treibers zur SPS. Verbindungszustände: ▸ *0:* Verbindung OK ▸ *1:* Verbindung gestört ▸ *2:* Verbindung simuliert Formatierung: **<Netzadresse>:<Verbindungszustand>;...;...;** Eine Verbindung ist erst nach dem ersten Anmelden einer Variablen bekannt. Damit eine Verbindung im String enthalten ist, muss einmal eine Variable dieser Verbindung angemeldet worden sein. Der Zustand einer Verbindung wird nur aktualisiert, wenn eine Variable der Verbindung angemeldet ist. Ansonsten wird nicht mit der entsprechenden Steuerung kommuniziert.

KONFIGURATION

Name aus Import	Typ	Offset	Erklärung
ReconnectInRead	*BOOL*	27	Wenn TRUE, dann wird beim Lesen automatisch ein Neuaufbau der Verbindung durchgeführt.
ApplyCom	*BOOL*	36	Änderungen an den Einstellungen der seriellen Schnittstelle zuweisen. Das

Name aus Import	Typ	Offset	Erklärung
			Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyCom zur Folge (aktuell ohne weitere Funktion).
ApplyModem	*BOOL*	37	Änderungen an den Modemeinstellungen zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyModem zur Folge. Diese schließt die aktuelle Verbindung und öffnet eine neue entsprechend den Einstellungen **PhoneNumberSet** und **ModemHwAdrSet**.
PhoneNumberSet	*STRING*	38	Telefonnummer, welche verwendet werden soll.
ModemHwAdrSet	*DINT*	39	Hardwareadresse, welche zu der Telefonnummer gehört.
GlobalUpdate	*UDINT*	3	Updatezeit in Millisekunden (ms).
BGlobalUpdaten	*BOOL*	4	TRUE, wenn die Updatezeit global ist.
TreiberSimul	*BOOL*	5	TRUE, wenn der Treiber in Simulation ist.
TreiberProzab	*BOOL*	6	TRUE, wenn das Prozessabbild gehalten werden soll.
ModemActive	*BOOL*	7	TRUE, wenn das Modem bei diesem Treiber aktiv ist.
Device	*STRING*	8	Name der seriellen Schnittstelle oder Name des Modem.
ComPort	*UINT*	9	Nummer der seriellen Schnittstelle.
Baudrate	*UDINT*	10	Baudrate der seriellen Schnittstelle.
Parity	*SINT*	11	Parität der seriellen Schnittstelle.
ByteSize	*USINT*	14	Bitanzahl pro Zeichen der seriellen Schnittstelle. Wert = *0*, wenn der Treiber keine serielle Kommunikation herstellen kann.
StopBit	*USINT*	13	Anzahl der Stoppbits der seriellen Schnittstelle.

Name aus Import	Typ	Offset	Erklärung
Autoconnect	*BOOL*	16	TRUE, wenn die Modemverbindung automatisch beim Lesen/Schreiben aufgebaut werden soll.
PhoneNumber	*STRING*	17	Aktuelle Telefonnummer.
ModemHwAdr	*DINT*	21	Hardwareadresse zur aktuellen Telefonnummer.
RxIdleTime	*UINT*	18	Wenn länger als diese Zeit in Sekunden (s) erfolgreich kein Datenverkehr stattfindet, wird die Modemverbindung beendet.
WriteTimeout	*UDINT*	19	Maximale Schreibdauer bei einer Modemverbindung in Millisekunden (ms).
RingCountSet	*UDINT*	20	So oft läutet ein hereinkommender Anruf, bevor dieser angenommen wird.
ReCallIdleTime	*UINT*	53	Wartezeit zwischen Anrufen in Sekunden (s).
ConnectTimeout	*UINT*	54	Zeit in Sekunden (s) für Verbindungsaufbau.

STATISTIK

Name aus Import	Typ	Offset	Erklärung
MaxWriteTime	*UDINT*	31	Längste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MinWriteTime	*UDINT*	32	Kürzeste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MaxBlkReadTime	*UDINT*	40	Längste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
MinBlkReadTime	*UDINT*	41	Kürzeste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
WriteErrorCount	*UDINT*	33	Anzahl der Schreibfehler.
ReadSucceedCount	*UDINT*	35	Anzahl der erfolgreichen Leseversuche.
MaxCycleTime	*UDINT*	22	Längste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
MinCycleTime	*UDINT*	23	Kürzeste Zeit in Millisekunden (ms), die zum Lesen

Name aus Import	Typ	Offset	Erklärung
			aller angeforderten Daten benötigt wurde.
WriteCount	*UDINT*	26	Anzahl der Schreibversuche.
ReadErrorCount	*UDINT*	34	Anzahl der fehlerhaften Leseversuche.
MaxUpdateTimeNormal	*UDINT*	56	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Normal** in Millisekunden (ms).
MaxUpdateTimeHigher	*UDINT*	57	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Höher** in Millisekunden (ms).
MaxUpdateTimeHigh	*UDINT*	58	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Hoch** in Millisekunden (ms).
MaxUpdateTimeHighest	*UDINT*	59	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Höchste** in Millisekunden (ms).
PokeFinish	*BOOL*	55	Geht für eine Abfrage auf *1*, wenn alle anstehenden Pokes ausgeführt wurden.

FEHLERMELDUNGEN

Name aus Import	Typ	Offset	Erklärung
ErrorTimeDW	*UDINT*	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler auftrat.
ErrorTimeS	*STRING*	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler als String auftrat.
RdErrPrimObj	*UDINT*	42	Nummer des PrimObjektes, als der letzte Lesefehler verursacht wurde.
RdErrStationsName	*STRING*	43	Name der Station, als der letzte Lesefehler verursacht wurde.
RdErrBlockCount	*UINT*	44	Anzahl der zu lesenden Blöcke, als der letzte Lesefehler verursacht wurde.
RdErrHwAdresse	*DINT*	45	Hardwareadresse, als der letzte Lesefehler verursacht wurde.
RdErrDatablockNo	*UDINT*	46	Bausteinnummer, als der letzte Lesefehler verursacht wurde.
RdErrMarkerNo	*UDINT*	47	Merkernummer, als der letzte Lesefehler verursacht wurde.

Name aus Import	Typ	Offset	Erklärung
RdErrSize	*UDINT*	48	Blockgröße, als der letzte Lesefehler verursacht wurde.
DrvError	*USINT*	25	Fehlermeldung als Nummer.
DrvErrorMsg	*STRING*	30	Fehlermeldung als Klartext.
ErrorFile	*STRING*	15	Name der Fehlerprotokolldatei.

8 Treiberspezifische Funktionen

Statusmeldungen und Analogwerte stellen treiberspezifische Funktionen dar.

Detaillierte Informationen hierzu finden Sie jeweils im entsprechenden Kapitel.

8.1 Statusmeldungen

Statusmeldungen werden via **ISP-IP CAMP** Protokoll kommuniziert.

- Nach erfolgreichem Kommunikationsaufbau zur SPS sendet der Treiber eine Summenabfrage an die Zentrale, mit der der Treiber kommuniziert. Diese Anfrage gilt für alle Elemente und alle verbundenen Zentralen.
- In der Antwort erhält der Treiber alle Statusmeldungen, die vom Normalzustand abweichen.
- Variablen vom Treiberobjekttyp *Status*, für die kein abweichender Zustand empfangen wurde, werden mit dem Hauptstatus *0* und dem Sub-Status *0* initialisiert (**IDLE**). Das entspricht dem Variablenwert *0*. Der Variablenstatus wird dabei auf *SPONT* gesetzt. Der Treiber implementiert ein Schatten-Abbild (= shadow image). Wird beim Start der Kommunikation ein abweichender Wert empfangen, wird dieser Wert im Schatten-Abbild gespeichert. Bei der erste Verwendung der Variable in der Runtime bekommt die Variable den abweichenden Wert aus dem Schatten-Abbild.
- Statusmeldungen sind bit-orientiert:
  - Die höherwertigen 8-Bits stellen den Hauptstatus dar.
  - Die niederwertigen 8-Bits stellen den Sub-Status dar.

  Die allgemeine Aktivierung eines Ausgangs wird somit mit dem dezimalen Wert *3072* dargestellt. Dies entspricht hexadezimal dem Wert *0x0C 00*.

Hauptstatus Aktivierung Ausgang: *dezimal 12*

Substatus allgemeine Aktivierung Ausgang: *dezimal 0*

8.2 Analogwerte

Analogwerte werden via ISP-IP AVP Protokoll kommuniziert.

- Nachdem die Kommunikation hergestellt wurde, aktiviert der Treiber die spontane Übertragung von Analogwerten. Dies gilt für alle Elemente aller verbundenen Zentralen.
- Die Analogwerte treffen nach und nach beim Treiber ein und werden im Schattenabbild gespeichert.
- Bei der ersten Verwendung in der Runtime bekommt die Variable den Wert aus dem Schattenabbild.
- Welche Analogwerte gesendet werden ist abhängig vom Elementtyp. Nicht alle Elementtypen unterstützen z.B. einen Wert für CO.

8.3 Kommandos

Für Kommandos gilt:

- Kommandos werden via **ISP-IP CAMP** Protokoll kommuniziert.
- Um ein Kommando an ein Element zu senden muss im zenon Projekt eine Kommandovariable mit entsprechende Adressierung erstellt sein.
- In der zenon Runtime wird eine Kommandovariable mit einem Idle-Wert von *0xFFFF* initialisiert.
- Das Kommando wird über das Schreiben des Kommandowertes auf die Variable initiiert.
- Auf Protokollebene kann ein erfolgreiches Kommando einerseits mit einer Antwort quittiert werden oder andererseits mit dem Statuswechsel des Elementes.
- Wenn keine Antwort auf ein Kommando kommt und kein Statuswechsel des Elements erfolgt, führt dies zu einer Zeitüberschreitung (= **Timeout**).
- Um das Ergebnis eines Kommandos abzubilden, wird der Status auf der Kommandovariable selbst abgebildet.

Ein Kommando an ein Element der Unterzentrale wird bei fehlender Berechtigung bereits von der Hauptzentrale gefiltert. Es erfolgt in diesem Fall keine negative Antwort von der Zentrale, es kommt zu einer Zeitüberschreitung bei der Befehlsausführung.

Sofern ein Kommando nicht zu einem Zusandswechsel führt oder es zu keinem (negativen) Befehlsquittierung kommt, soll geprüft werden ob das Element das entsprechende Kommando unterstützt und ob die Berechtigung für das Leitsystem in das Berechtigungsmakro in der

Hauptzentrale vorhanden ist. Für die Unterzentrale muss das Berechtigungsmakro der Hauptzentrale die Berechtigung entsprechend enthalten.

8.3.1 Sollwert für Kommando

Ein Sollwert für ein Kommando setzt sich aus der Kommando-Nummer und dem Kommando-Wert (*Validity*) zusammen. Die Kommando-Nummer belegt dabei die höherwertigen Bytes. Der Kommando-Wert (*Validity*) belegt die niederwertigen Bytes. Nicht alle Kommando-Kombinationen sind gültig.

Die möglichen Kombinationen von Kommando-Nummern und den dazugehörigen Kommando-Werten finden Sie im Kapitel Unterstützte Befehle (auf Seite 43). Nicht alle Kommandos werden von allen Elementen unterstützt.

BEISPIEL 1

ElementTypeClass Output, FunctionType Self, Off, General

Sollwert für die Kommandovariable: 0x0000

CommandNumber: *Off (0)*

Validity: *Abschalten (0)*

BEISPIEL 2

ElementTypeClass Output, FunctionType Self, On, Acoustic

Sollwert für die Kommandovariable: 0x0101

CommandNumber: *On (1)*

Validity: *Akustik einschalten (1)*

8.3.2 Unterstützte Befehle

Der **ISPIP Treiber** unterstützt Kommando-Kombinationen gemäß der **N3 CAMP** Spezifikation. Beachten Sie, dass nicht alle Elementtypen alle Kommandos unterstützen.

Informationen zur jeweiligen **ElementTypeClass** finden Sie in der entsprechenden Tabelle. Die jeweilige Kommandonummer und Validitiy Nummer sind in der Tabelle aufgenommen.

Beachten Sie, dass die Validity **Acoustic** einen unterschiedlichen Wert hat, abhängig von der Kommandonummer.

Kommandowert für Zone, **FunctionType Self**, **Off**, **General**: 0x0000 (Dezimal 0)

Kommandowert für Zone, **FunctionType Self**, **On**, **Emergency**: 0x0103 (Dezimal 259)

ZONE

FunctionType	CommandNumber	Validity
Self	**Off (0)**	*General (0)* *Emergency (3)* *Limited (5)*
	On (1)	*General (0)* *Emergency (3)* *AtTime (5)*
	Reset (3)	*General (0)* *Alarm (1)*
	Revision (4)	*General (0)*
	Check (5)	*General (0)*
	SimulateAlarm (6)	*General (0)*
	SimulateFault (7)	*General (0)*
	OffInternal (9)	*General (0)*
	SilentRevision (17)	*General (0)*
	RevisionCheck (18)	*General (0)*
Pollution	**Reset (3)**	*General (0)*

INPUT

FunctionType	CommandNumber	Validity
Self	**Off (0)**	*General (0)*
	On (1)	*General (0)*
	Reset (3)	*General (0)*
	Revision (4)	*General (0)*
	SimulateFault (7)	*General (0)*

FunctionType	CommandNumber	Validity
	SimulateActive (8)	*General (0)*
	SimulatePreActive (16)	*General (0)*
	SilentRevision (17)	*General (0)*

OUTPUT

FunctionType	CommandNumber	Validity
Self	**Off (0)**	*General (0)* *Acoustic (1)* *FireProtectionEquipment (2)*
	On (1)	*General (0)* *Acoustic (1)* *FireProtectionEquipment (2)*
	Set (2)	*General (0)* *Critical (1)*
	Reset (3)	*General (0)* *Acoustic (2)*
	Revision (4)	*General (0)*
	SimulateFault (7)	*General (0)*
	Reactivate (10)	*General (0)* *Acoustic (1)*
Acknowledge	-	-

EXTERNAL

FunctionType	CommandNumber	Validity
Self	**Off (0)**	*General (0)* *Emergency (3)*
	On (1)	*General (0)*

FunctionType	CommandNumber	Validity
		Emergency (3)
	Set (2)	*General (0)*
	Reset (3)	*General (0)* *Alarm (1)*
	Revision (4)	*General (0)*
	SimulateAlarm (6)	*General (0)*
	SimulateFault (7)	*General (0)*

PRINTER

FunctionType	CommandNumber	Validity
Self	**Off (0)**	*General (0)* *PaperFeed (4)*
	On (1)	*General (0)* *PaperFeed (4)*
	Reset (3)	*General (0)* *PageNumber (3)* *MessageNumber (4)*
	Check (5)	*General (0)*
	Init (11)	*General (0)*
	TopOfPage (12)	*General (0)*
RangeFilter	**Off (0)**	*General (0)*
	On (1)	*General (0)*
MessageFilter	**Off (0)**	*General (0)*
	Level (13)	*0-3*

CI

FunctionType	CommandNumber	Validity
Self	-	-
RangeFilter	-	-
Operation	-	-
Acoustic	Reset (3)	*General (0)*

BATTERY

FunctionType	CommandNumber	Validity
Self	Reset (3)	*General (0)*
	Check (5)	*General (0)*

NET

FunctionType	CommandNumber	Validity
Self	Reset (3)	*General (0)*

MODULEACTIVE

FunctionType	CommandNumber	Validity
Self	-	-

MODULEPASSIVE

FunctionType	CommandNumber	Validity
Self	-	-

DELAYLAYER

FunctionType	CommandNumber	Validity
Self	Off (0)	*General (0)*
	On (1)	*General (0)*
Automatic	Off (0)	*General (0)*
	On (1)	*General (0)*

FBP

FunctionType	CommandNumber	Validity
Self	-	-
Acoustic	Reset (3)	*General (0)*

SCU

FunctionType	CommandNumber	Validity
Self	-	-

INTERVENTION

FunctionType	CommandNumber	Validity
Self	Reset (3)	*General (0)*

CONNECTION

FunctionType	CommandNumber	Validity
Self	-	-

MASTERSYSTEM

FunctionType	CommandNumber	Validity
Self	Off (0)	*General (0)*
	On (1)	*General (0)*
	Reset (3)	*General (0)*
Operation	-	-
LocalOperation	-	-

LOOP

FunctionType	CommandNumber	Validity
Self	Off (0)	*General (0)*
	On (1)	*General (0)*
	Reset (3)	*General (0)*

FunctionType	CommandNumber	Validity
	Revision (4)	*General (0)*
	Check (5)	*General (0)*
	StartupStop (15)	*General (0)*
Logic	Mapping (14)	*General (0)*

FLOODINGZONE

FunctionType	CommandNumber	Validity
Self	Off (0)	*General (0)*
	On (1)	*General (0)*
	Reset (3)	*General (0)* *Acoustic (1)*
Automatic	Off (0)	*General (0)*
	On (1)	*General (0)*
RevisionOperation	Set (2)	*General (0)*
	Reset (3)	*General (0)*

REMOTEACCESS

FunctionType	CommandNumber	Validity
Self	Set (2)	*General (0)*

ALARMAREA

FunctionType	CommandNumber	Validity
Self	Off (0)	*General (0)* *Acoustic (1)*
	On (1)	*General (0)* *Acoustic (1)*
	Set (2)	*General (0)*

FunctionType	CommandNumber	Validity
	Reset (3)	*General (0)* *Acoustic (2)*
	Revision (4)	*General (0)*
	SimulateFault (7)	*General (0)*
	Reactivate (10)	*General (0)* *Acoustic (1)*

SUPERORDINATEDCONTROLUNIT

FunctionType	CommandNumber	Validity
Self	-	-

SUBORDINATEDCONTROLUNIT

FunctionType	CommandNumber	Validity
Self	-	-

NETWORK

FunctionType	CommandNumber	Validity
Self	-	-

EXTERNALSYSTEM

FunctionType	CommandNumber	Validity
Self	**Off (0)**	*General (0)*
	On (1)	*General (0)*
	Reset (3)	*General (0)*
	Check (5)	*General (0)*
LocalOperation	-	-

8.3.3 Treiberspezifische Funktionen

RÜCKGABEWERTE NACH KOMMANDOAUSFÜHRUNG

Variablenwert	Beschreibung
0xFFFF	Idle
0xFFFE	Timeout nach der Ausführung.
0xFFFD	Kommando gesendet.
0xFFFB	Kommando konnte nicht erfolgreich gesendet werden.
0xFFF9	Negative Antwort auf das Kommando. Der Befehl soll mittels SetCritical ausgeführt werden.
0xFFF8	Negative Antwort auf das Kommando: die Hauptzentrale sendet eine negative Antwort für die Unterzentrale.
0xFFF7	Negative Antwort auf Batterietest. Führen sie den Batterietest erneut aus.
0xFFF6	Negative Antwort auf das Kommando. Das Element ist zur Zeit nicht verfügbar (Hardwaremodul ausgefallen).
0xFFF5	Negative Antwort auf Anfrage: das Element unterstützt eine Einzelabfrage nicht.
0xFFF4	Negative Antwort: das adressierte Element existiert nicht.
0xFFF3	Negative Antwort: das Kommando ist für ein Sub Element nicht möglich oder unzulässig.
0xFFF2	Negative Antwort: unpassendes Kommando für den Elementzustand.
0xFFF1	Negative Antwort: fehlende Berechtigung
0xFF00	Positive Rückmeldung auf das Kommando

9 Funktion Treiberkommandos

Die zenon Funktion **Treiberkommandos** dient dazu, Treiber über zenon zu beeinflussen.
Mit einem Treiberkommando können Sie einen Treiber:

- starten

- stoppen
- in einen bestimmten Treibermodus versetzen
- zu bestimmten Aktionen veranlassen

Hinweis: Dieses Kapitel beschreibt Standardfunktionalitäten, die für die meisten zenon Treiber gültig sind.
Nicht alle hier beschriebenen Funktionalitäten stehen für jeden Treiber zur Verfügung. Zum Beispiel enthält ein Treiber, der laut Datenblatt keine Modemverbindung unterstützt, auch keine Modem-Funktionalitäten.

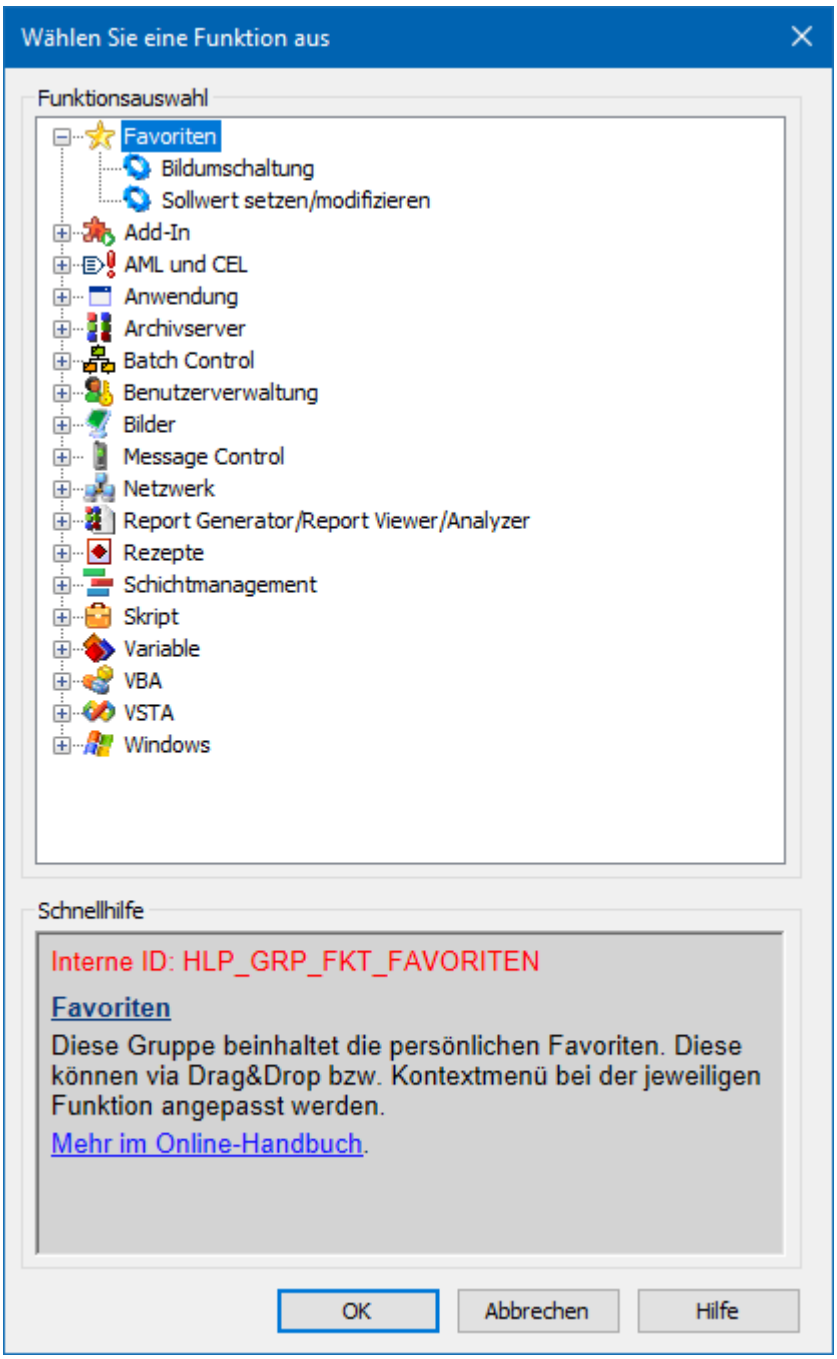
> **Achtung**
>
> Die zenon Funktion **Treiberkommandos** ist nicht ident mit den Treiberkommandos, die bei Energy-Treibern in der Runtime ausgeführt werden können!

PROJEKTIERUNG DER FUNKTION

Die Projektierung erfolgt über die Funktion **Treiberkommandos**.
Um die Funktion zu projektieren:

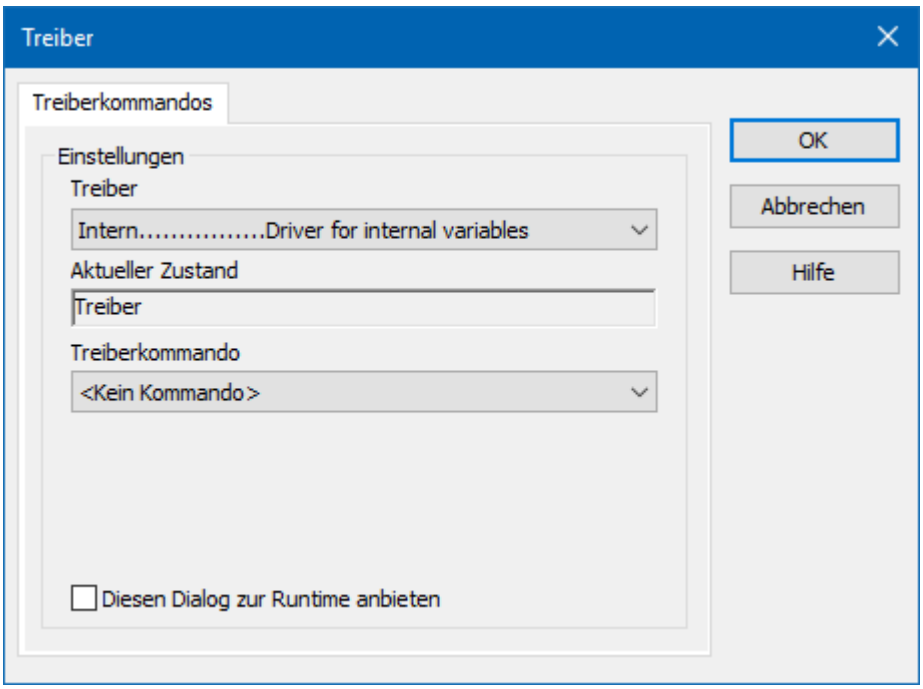
1. Legen Sie im zenon Editor eine neue Funktion an.

Der Dialog zur Auswahl einer Funktion wird geöffnet.



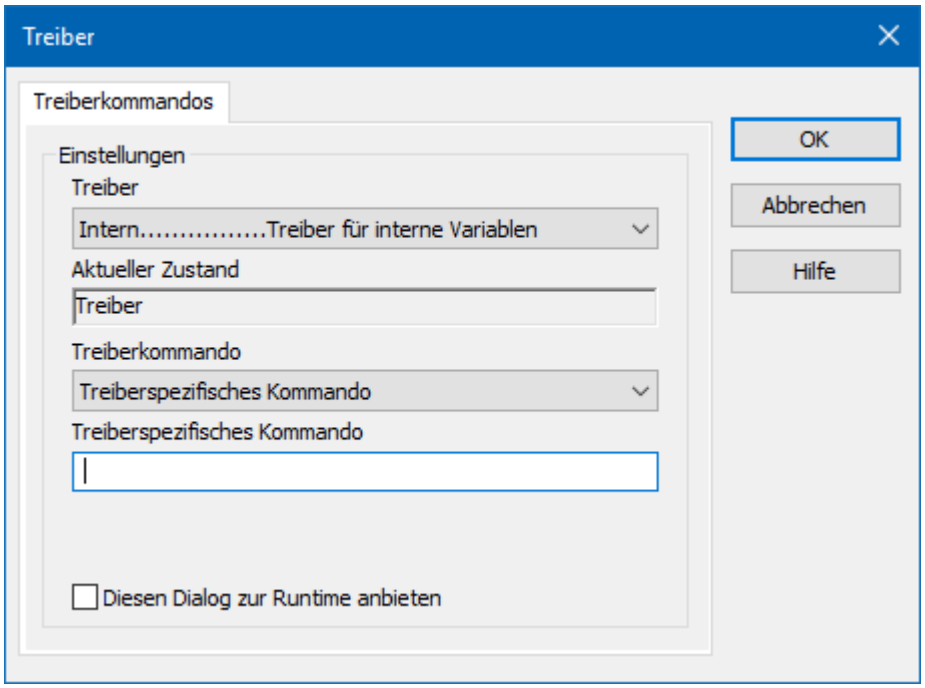
2. Navigieren Sie zum Knoten **Variable.**
3. Wählen Sie den Eintrag **Treiberkommandos**.

Der Dialog zur Konfiguration wird geöffnet.



4. Wählen Sie den gewünschten Treiber und das benötigte Kommando aus.
5. Schließen Sie den Dialog mit Klick auf **OK** und stellen Sie sicher, dass die Funktion in der Runtime ausgeführt wird.
   Beachten Sie die Hinweise im Abschnitt **Funktion Treiberkommandos im Netzwerk**.

DIALOG TREIBERKOMMANDOS



Option	Beschreibung
Treiber	Auswahl des Treibers aus der Dropdownliste. Diese enthält alle im Projekt geladenen Treibern.
Aktueller Zustand	Fixer Eintrag, der vom System gesetzt wird. In aktuellen Versionen ohne Funktion.
Treiberkommando	Auswahl des gewünschten Treiberkommandos aus Dropdownliste. Details zu den konfigurierbaren Treiberkommandos siehe Abschnitt **Verfügbare Treiberkommandos**.
Treiberspezifisches Kommando	Eingabe eines für den gewählten Treiber spezifischen Kommandos. **Hinweis:** Nur verfügbar, wenn für die Option **Treiberkommando** der Eintrag *Treiberspezifisches Kommando* gewählt wurde.
Diesen Dialog zur Runtime anbieten	Konfiguration, ob die Konfiguration in der Runtime geändert werden kann: ▸ *Aktiv*: Dieser Dialog wird in der Runtime vor dem Ausführen der Funktion geöffnet. Die Konfiguration kann damit in der Runtime vor der Ausführung noch geändert werden. ▸ *Inaktiv*: Die Editor-Konfiguration wird in der Runtime beim Ausführen der Funktion angewendet. Default: *inaktiv*

DIALOG BEENDEN

Option	Beschreibung
OK	Übernimmt Einstellungen und schließt den Dialog.
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

VERFÜGBARE TREIBERKOMMANDOS

Diese Treiberkommandos stehen - abhängig vom gewählten Treiber - zur Verfügung:

Treiberkommando	Beschreibung
Kein Kommando	Es wird kein Kommando gesendet. Damit kann auch ein bereits bestehendes Kommando aus einer projektierten Funktion entfernt werden.
Treiber starten (Onlinemodus)	Treiber wird neu initialisiert und gestartet. **Hinweis:** Wenn der Treiber bereits gestartet wurde, muss er gestoppt werden. Erst dann kann der Treiber wieder neu initialisiert und gestartet werden.
Treiber stoppen (Offlinemodus)	Treiber wird angehalten, es werden keine neuen Daten angenommen. **Hinweis:** Ist der Treiber im Offline-Modus, erhalten alle Variablen, die für diesem Treiber angelegt wurden, den Status *Abgeschaltet* (*OFF*; Bit *20*).
Treiber in Simulationsmodus	Treiber wird in den Simulationsmodus gesetzt. Die Werte aller Variablen des Treibers werden vom Treiber simuliert. Es werden keine Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
Treiber in Hardwaremodus	Treiber wird in den Hardwaremodus gesetzt. Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
Treiberspezifisches Kommando	Eingabe eines treiberspezifischen Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.
Treiber Sollwertsetzen aktivieren	Sollwert setzen auf Treiber ist möglich.
Treiber Sollwertsetzen deaktivieren	Sollwert setzen auf Treiber wird verhindert.
Verbindung mit Modem aufbauen	Verbindung aufbauen (für Modem-Treiber). Öffnet Eingabefelder für Hardware-Adresse und Eingabe der zu wählenden Nummer.
Verbindung mit Modem trennen	Verbindung beenden (für Modem-Treiber).
Treiber in Simulationsmodus zählend	Treiber wird in den zählenden Simulationsmodus gesetzt. Alle Werte werden mit *0* initialisiert und in der eingestellten Updatezeit jeweils um *1* bis zum Maximalwert inkrementiert und beginnen dann wieder

Treiberkommando	Beschreibung
	bei *0*.
Treiber in Simulationsmodus statisch	Es wird keine Kommunikation zur Steuerung aufgebaut. Alle Werte werden mit *0* initialisiert.
Treiber in Simulationsmodus programmiert	Die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in der zenon Logic Runtime ab.

FUNKTION TREIBERKOMMANDOS IM NETZWERK

Wenn sich der Rechner, auf dem die Funktion **Treiberkommandos** ausgeführt wird, im zenon Netzwerk befindet, werden zusätzlich weitere Aktionen ausgeführt:

- Ein spezielles Netzwerkkommando wird vom Rechner zum Server des Projekts gesendet. Dieser führt dann die gewünschte Aktion auf seinem Treiber aus.
- Zusätzlich sendet der Server das gleiche Treiberkommando zum Standby des Projekts. Der Standby führt die Aktion auch auf seinem Treiber aus.

Dadurch ist gewährleistet, dass Server und Standby synchronisiert sind. Dies funktioniert nur, wenn Server und Standby jeweils eine funktionierende und unabhängige Verbindung zur Hardware haben.

10 Fehleranalyse

Sollte es zu Kommunikationsproblemen kommen, bietet dieses Kapitel Hilfe, um den Fehler zu finden.

10.1 Analysetool

Alle zenon Module wie z. B. Editor, Runtime, Treiber, usw. schreiben Meldungen in eine gemeinsame LOG-Datei. Um sie korrekt und übersichtlich anzuzeigen, benutzen Sie das Programm Diagnosis Viewer, das mit zenon mitinstalliert wird. Sie finden es unter **Start/Alle Programme/zenon/Tools 8.20 -> Diagviewer.**

zenon Treiber protokollieren alle Fehler in LOG-Dateien. LOG-Dateien sind Textdateien mit einer speziellen Struktur. Der Standardordner für die LOG-Dateien ist der Ordner **LOG** unterhalb des Ordners **ProgramData**, zum Beispiel:

%ProgramData%\COPA-DATA\LOG.

Achtung: Mit den Standardeinstellungen zeichnet ein Treiber nur Fehlerinformationen auf. Mit dem Diagnosis Viewer kann bei den meisten Treibern die Diagnose-Ebene auf „Debug" und „Deep Debug"

erweitert werden. Damit protokolliert der Treiber auch alle anderen wesentlichen Aufgaben und Ereignisse.

Im Diagnosis Viewer kann man auch:

- neu erstellte Einträge in Echtzeit mitverfolgen
- die Aufzeichnungseinstellungen anpassen
- den Ordner, in dem die LOG-Dateien gespeichert werden, ändern

Hinweise:

1. Der Diagnosis Viewer zeigt alle Einträge in UTC (Koordinierter Weltzeit) an und nicht in der lokalen Zeit.
2. Der Diagnosis Viewer zeigt in seiner Standardeinstellung nicht alle Spalten einer LOG-Datei an. Um mehr Spalten anzuzeigen, aktivieren Sie die Eigenschaft **Add all columns with entry** im Kontextmenü der Spaltentitel.
3. Bei Verwendung von reinem **Error-Logging** befindet sich eine Problembeschreibung in der Spalte **Error text**. In anderen Diagnose-Ebenen befindet sich diese Beschreibung in der Spalte **General text**.
4. Viele Treiber zeichnen bei Kommunikationsprobleme auch Fehlernummern auf, die die SPS ihnen zuweist. Diese werden in **Error text** und/oder **Error code** und/oder **Driver error parameter(1** und **2)** angezeigt. Hinweise zur Bedeutung der Fehlercodes erhalten Sie in der Treiberdokumentation und der Protokoll/SPS-Beschreibung.
5. Stellen Sie am Ende Ihrer Tests den Diagnose-Level von **Debug** oder **Deep Debug** wieder zurück. Bei **Debug** und **Deep Debug** fallen beim Protokollieren sehr viele Daten an, die auf der Festplatte gespeichert werden und die Leistung Ihres Systems beeinflussen können. Diese werden auch nach dem Schließen des Diagnosis Viewers weiter aufgezeichnet.

> **Achtung**
>
> Unter Windows CE werden aus Ressourcegründen Fehler standardmäßig nicht protokolliert.

Weitere Informationen zum Diagnosis Viewer finden Sie im Handbuch Diagnosis Viewer.

10.2 Treiberüberwachung

Die Runtime überwacht die Verfügbarkeit des Treibers via Watchdog. Ist ein Treiber nicht mehr verfügbar, wird für alle angemeldeten Variablen des Treibers zusätzlich das Statusbit *INVALID* gesetzt.

Mögliche Ursachen für Auslösen des Watchdogs:

- Der Treiberprozess läuft nicht mehr.

  Überprüfen Sie im Task Manager ob die Treiber-Exe noch läuft.

- Betriebssystem ist mit höher priorisierten Prozessen ausgelastet.

  Überprüfen Sie die Konfiguration Ihres Systems auf zu wenig Arbeitsspeicher und zu geringe CPU-Leistung. In diesem Fall erfolgt das Rücksetzen des *INVALID* Statusbits durch den Treiber nur bei Wertänderung auf der Gegenstelle. Statische Werte behalten das *INVALID* Statusbit bis zum nächsten Start der Runtime oder des Treibers.

WATCHDOG KONFIGURATION

Für die Überwachung der Kommunikation zur Runtime wird die Verbindung zum Treiber in einem fix vorgegebenen Zeitraum von 60 Sekunden überprüft. Dieser Vorgang wird mehrmals wiederholt. Konnte nach 5 Versuchen (= innerhalb von 5 Minuten) keine valide Verbindung zum Treiber erkannt werden, wird das *INVALID*-Bit bei der angemeldeten (*advised*) Variablen gesetzt. Zusätzlich wird das *INVALID*-Bit auch gesetzt, wenn neue Variablen angemeldeten werden. Das *INVALID*-Bit wird nicht mehr zurückgesetzt.

Dafür werden entsprechende LOG-Einträge erstellt.

LOG-EINTRAG

Bei Auslösen des Watchdogs wird eine Fehlermeldung im LOG protokolliert:

Parameter	Beschreibung
Communication with driver:<drvExe>/<drvDesc>(id:<drvId>) timed out. No communication for <time> ms.	Keine Kommunikation mit Treiber innerhalb der angegeben Zeit. - *<time>*: Zeitangabe in Millisekunden - *<drvDesc>*: Treibername - *<drvExe>*: Treiber EXE-Name - *<drvId>*: Treiber-ID im zenon Projekt
Communication with %s timed out. Invalid-Bit will be set.	Die Kommunikation zum Treiber *%s* konnte bei 5 Versuchen nicht innerhalb von 60 Sekunden aufgebaut werden. Bei der Variable wird das *INVALID*-Bit gesetzt.
Communication with %s timed out. Timeout happened %d times	Die Kommunikation zum Treiber *%s* konnte *%d* Mal nicht innerhalb von 60 Sekunden aufgebaut werden.

10.3 Checkliste

ALLGEMEINE FEHLERSUCHE

Überprüfen Sie bei Fehlern:

- Ist die Steuerung an die Stromversorgung angeschlossen?
- Analyse mit Hilfe des **Diagnose Viewers** (auf Seite 57):
  -> Welche Meldungen werden angezeigt?
- Sind die Teilnehmer im **TCP/IP**-Netz verfügbar?
- Kann die Steuerung über den Befehl *Ping* erreicht werden?

  Ping: **Kommandozeile öffnen -> ping <IP-Adresse> (z. B.: ping 192.168.0.100) -> Taste Eingabe drücken.**

  Kommt eine Antwort mit Zeitangabe oder ein Timeout?
- Kann die Steuerung auf dem entsprechenden Port über *Telnet* erreicht werden?

  Telnet: **Kommandozeile öffnen, telnet <IP-Adresse Port-Nummer> eingeben (z. B. für Modbus: telnet 192.168.0.100 502) -> Taste Eingabe drücken .**

  Wird der Bildschirm schwarz, konnte eine Verbindung aufgebaut werden.
- Wurde die Netzadresse in den Adresseigenschaften der Variable korrekt eingestellt?
  - Stimmt die Adressierung mit der Konfiguration im Treiberdialog überein?
  - Entspricht die Netzadresse der Adresse der Zielstation?
- Wird in der Variable der richtige Objekttyp verwendet?

  Beispiel: Treibervariablen sind reine Statistikvariablen und kommunizieren nicht mit der Steuerung. (Siehe Kapitel Treibervariablen (auf Seite 35).)
- Stimmt die Adressierung der Variable mit der in der Steuerung überein?
- Ist in der Anlage entsprechend ein Fremdleitsystem konfiguriert?
  - Für den Redundanzbetrieb müssen in der Anlage zwei Fremdleitsysteme konfiguriert sein. Einmal für den Server und einmal für den Standby Server
- Stimmt der Benutzername und das Passwort überein mit der Konfiugration in der Anlage?

zenon Treiber protokollieren alle Fehler in LOG-Dateien. LOG-Dateien sind Textdateien mit einer speziellen Struktur. Der Standardordner für die LOG-Dateien ist der Ordner **LOG** unterhalb des Ordners **ProgramData**, zum Beispiel:

%ProgramData%\COPA-DATA\LOG.

Achtung: Mit den Standardeinstellungen zeichnet ein Treiber nur Fehlerinformationen auf. Mit dem Diagnosis Viewer kann bei den meisten Treibern die Diagnose-Ebene auf „Debug" und „Deep Debug"

erweitert werden. Damit protokolliert der Treiber auch alle anderen wesentlichen Aufgaben und Ereignisse. Weitere Informationen zum Diagnosis Viewer finden Sie im Handbuch Diagnosis Viewer.

Für die weitere Fehleranalyse werden benötigt:

- die Projektsicherung
- LOG-Dateien

Senden Sie diese nach Rücksprache mit dem Kundendienst an Ihren zuständigen Support.

MANCHE VARIABLEN MELDEN INVALID

- INVALID Bits beziehen sich immer auf eine Netzadresse.
- Bei fehlerhafte Addressierung für State Variablen gibt es kein INVALID Statusbit

WERTE WERDEN NICHT ANGEZEIGT, ZAHLENWERTE BLEIBEN LEER

Treiber läuft nicht. Überprüfen Sie die:

- Installation von zenon
- Installation des Treibers
- Installation aller Komponenten
  -> Achten Sie auf Fehlermeldungen beim Start der Runtime.

VARIABLEN WERDEN MIT BLAUEM PUNKT ANGEZEIGT

Die Kommunikation im Netzwerk ist gestört:

- Bei einem Netzwerkprojekt:
  Läuft das Netzwerkprojekt auch auf dem Server?
- Bei einem Standalone-Projekt oder Netzwerkprojekt, das auf dem Server läuft:
  Deaktivieren Sie die Eigenschaft **Nur von Standby Server anfordern** im Knoten **Treiber Anbindung/Adressierung**.

WERTE WERDEN FALSCH ANGEZEIGT

Überprüfen Sie die Angaben zur Berechnung im Knoten **Wertberechnung** der Variablen-Eigenschaften?

TREIBER FÄLLT SPORADISCH AUS

Analyse mit Hilfe des **Diagnose Viewers** (auf Seite 57):
-> Welche Meldungen werden angezeigt?